

4

Performance Enhancements Made with sQl server 2014

By PInal Dave DeveloPer evangelIst

SQL Server Whitepaper



Contents

sQl server 2014: Columnstore enhancements ....................................................... 3

enhancements to ssMs .......................................................................................3

Columnstore Query Processing .......................................................................... 4

Clustered Columnstore syntax ............................................................................ 4

Columnstore, Partitioning and Data archival .................................................5

sQl server 2014: Managed lock Priority ...................................................................6

tsQl extensions for Managed lock Priority ...................................................6

experiencing Managed lock Priority ................................................................. 7

Buffer pool extension ........................................................................................................8

Why this feature? ......................................................................................................8

sQl server 2014: seleCt Into is truly Parallel ...........................................9

Final Words ......................................................................................................................... 10

Solution Overview

It is second to human nature to question and get the best out of everything. It is in our DNA to see how we can maximize on our current investments and if our investments are reaping great benefits. If you are buying a brand new car, you are likely to look at multiple dimensions like cost, looks, reviews, spacious, utility and finally on the mileage too. The fundamental definition of mileage is – “The ratio of the number of miles travelled to the number of gallons of petrol burned”, in essence we are looking at the performance of the vehicle.

These analogies are great not only in our personal life but also in professional organizations. CIOs and CFOs are questioning the investments being done on software’s everywhere. Be it investing in cloud or upgrade to a new version of software. The question always remains WHY? With the release of SQL Server 2014, these questions also come in the mind of a typical DBA and a future work item to upgrade his current infrastructure. It is not enough to just do a upgrade but also utilize the best features of the next version post upgrade to improve manageability, maintainability and if we can give better performance to application – then the upgrade process is worth every penny and time spent.

In this article we will take a quick tour on our Top 4 features introduced with SQL Server 2014 from a performance point of view. The first enhancement will be on Columnstore Index.



SQL Server 2014: Columnstore Enhancements

SQL Server 2012 introduced a new feature called Non-Clustered ColumnStore Index which allowed storage of SQL Server data in a column-based storage and column-based processing. It was a great addition to SQL Server arsenal and DBA’s could make use of this feature for data warehousing workloads and increase performance of read-heavy workloads. The basics of using a columnstore index for SQL Server 2012 is described extensively on MSDN. There are a number of restrictions in the previous implementation as described on MSDN and the most important limitation remained we cannot update a table which has a columnstore index. This limited the use of this feature in most of the application except those having read-only data warehousing queries.

In this article let us take a tour on some of the enhancements that were introduced with SQL Server 2014 in this area. The Columnstore index can either be Clustered or NonClustered. The limitations and restrictions for NonClustered Index remain the same as in previous versions. Below are some of the salient points of the new Clustered Columnstore indexes.

1. New columnstore is updateable

2. Has all the columns of the table and stores the whole table as columnar

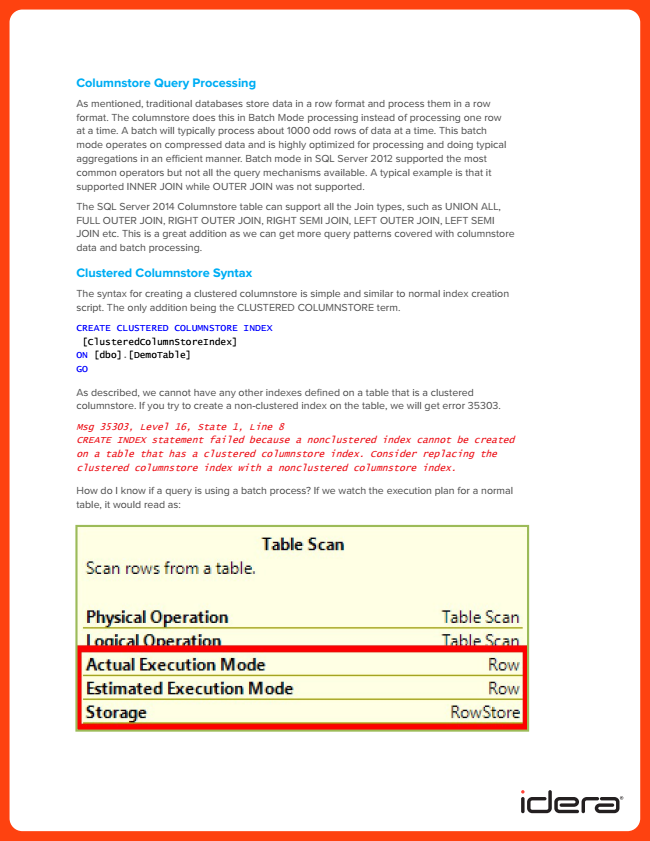
3. Uses columnstore compression out-of-box

4. This is the only index that can be created on the table

**Enhancements to SSMS**

A table with the new Clustered Columnstore will have the following new icon in Management Studio like:

SQL Server Management Studio has enhanced the shell context menu, where we can create columnstore indexes; both clustered and non-clustered.



**Columnstore Query Processing**

As mentioned, traditional databases store data in a row format and process them in a row format. The columnstore does this in Batch Mode processing instead of processing one row at a time. A batch will typically process about 1000 odd rows of data at a time. This batch mode operates on compressed data and is highly optimized for processing and doing typical aggregations in an efficient manner. Batch mode in SQL Server 2012 supported the most common operators but not all the query mechanisms available. A typical example is that it supported INNER JOIN while OUTER JOIN was not supported.

The SQL Server 2014 Columnstore table can support all the Join types, such as UNION ALL, FULL OUTER JOIN, RIGHT OUTER JOIN, RIGHT SEMI JOIN, LEFT OUTER JOIN, LEFT SEMI JOIN etc. This is a great addition as we can get more query patterns covered with columnstore data and batch processing.

**Clustered Columnstore Syntax**

The syntax for creating a clustered columnstore is simple and similar to normal index creation script. The only addition being the CLUSTERED COLUMNSTORE term.

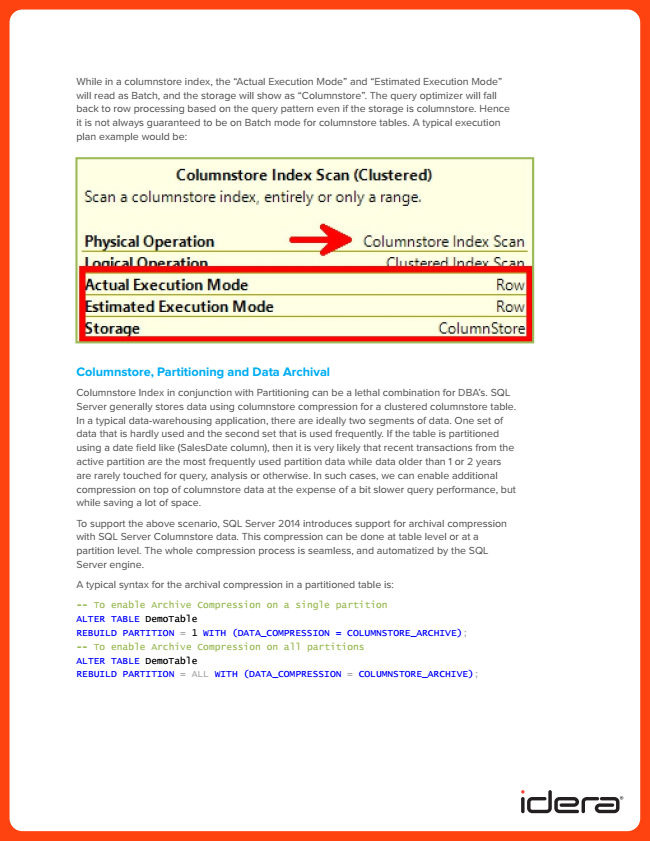
CREATE CLUSTERED COLUMNSTORE INDEX

[ClusteredColumnStoreIndex] ON [dbo].[DemoTable] GO

As described, we cannot have any other indexes defined on a table that is a clustered columnstore. If you try to create a non-clustered index on the table, we will get error 35303.

Msg 35303, Level 16, State 1, Line 8 CREATE INDEX statement failed because a nonclustered index cannot be created on a table that has a clustered columnstore index. Consider replacing the clustered columnstore index with a nonclustered columnstore index.

How do I know if a query is using a batch process? If we watch the execution plan for a normal table, it would read as:



While in a columnstore index, the “Actual Execution Mode” and “Estimated Execution Mode” will read as Batch, and the storage will show as “Columnstore”. The query optimizer will fall back to row processing based on the query pattern even if the storage is columnstore. Hence it is not always guaranteed to be on Batch mode for columnstore tables. A typical execution plan example would be:

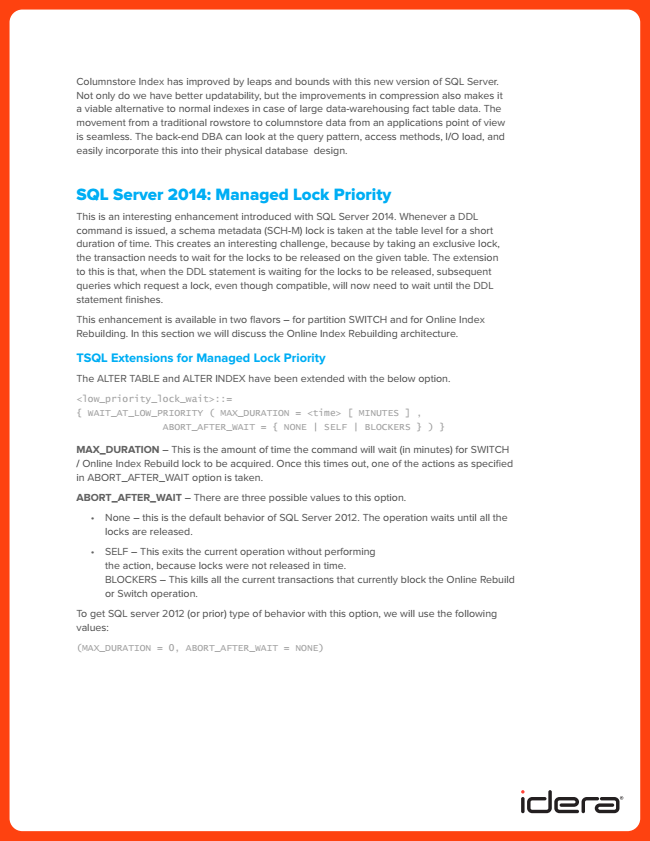
**Columnstore, Partitioning and Data Archival**

Columnstore Index in conjunction with Partitioning can be a lethal combination for DBA’s. SQL Server generally stores data using columnstore compression for a clustered columnstore table. In a typical data-warehousing application, there are ideally two segments of data. One set of data that is hardly used and the second set that is used frequently. If the table is partitioned using a date field like (SalesDate column), then it is very likely that recent transactions from the active partition are the most frequently used partition data while data older than 1 or 2 years are rarely touched for query, analysis or otherwise. In such cases, we can enable additional compression on top of columnstore data at the expense of a bit slower query performance, but while saving a lot of space.

To support the above scenario, SQL Server 2014 introduces support for archival compression with SQL Server Columnstore data. This compression can be done at table level or at a partition level. The whole compression process is seamless, and automatized by the SQL Server engine.

A typical syntax for the archival compression in a partitioned table is:

-- To enable Archive Compression on a single partition ALTER TABLE DemoTable REBUILD PARTITION = 1 WITH (DATA\_COMPRESSION = COLUMNSTORE\_ARCHIVE); -- To enable Archive Compression on all partitions ALTER TABLE DemoTable REBUILD PARTITION = ALL WITH (DATA\_COMPRESSION = COLUMNSTORE\_ARCHIVE);



Columnstore Index has improved by leaps and bounds with this new version of SQL Server. Not only do we have better updatability, but the improvements in compression also makes it a viable alternative to normal indexes in case of large data-warehousing fact table data. The movement from a traditional rowstore to columnstore data from an applications point of view is seamless. The back-end DBA can look at the query pattern, access methods, I/O load, and easily incorporate this into their physical database design.

SQL Server 2014: Managed Lock Priority

This is an interesting enhancement introduced with SQL Server 2014. Whenever a DDL command is issued, a schema metadata (SCH-M) lock is taken at the table level for a short duration of time. This creates an interesting challenge, because by taking an exclusive lock, the transaction needs to wait for the locks to be released on the given table. The extension to this is that, when the DDL statement is waiting for the locks to be released, subsequent queries which request a lock, even though compatible, will now need to wait until the DDL statement finishes.

This enhancement is available in two flavors – for partition SWITCH and for Online Index Rebuilding. In this section we will discuss the Online Index Rebuilding architecture.

**TSQL Extensions for Managed Lock Priority**

The ALTER TABLE and ALTER INDEX have been extended with the below option.

<low\_priority\_lock\_wait>::= { WAIT\_AT\_LOW\_PRIORITY ( MAX\_DURATION = <time> [ MINUTES ] ,

ABORT\_AFTER\_WAIT = { NONE | SELF | BLOCKERS } ) }

MAX\_DURATION – This is the amount of time the command will wait (in minutes) for SWITCH / Online Index Rebuild lock to be acquired. Once this times out, one of the actions as specified in ABORT\_AFTER\_WAIT option is taken.

ABORT\_AFTER\_WAIT – There are three possible values to this option.

• None – this is the default behavior of SQL Server 2012. The operation waits until all the locks are released.

• SELF – This exits the current operation without performing the action, because locks were not released in time. BLOCKERS – This kills all the current transactions that currently block the Online Rebuild or Switch operation.

To get SQL server 2012 (or prior) type of behavior with this option, we will use the following values:

(MAX\_DURATION = 0, ABORT\_AFTER\_WAIT = NONE)



**Experiencing Managed Lock Priority**

In this example we will be using a basic database with a table called MLP\_OIR. Our demo will include two phases.

1. Session 1: Start a transaction and make an SELECT

2. Session 2: Perform normal Online Rebuilding of Index using Alter statement.

3. The rebuilding operation must now be in blocking mode

-- Session 1 BEGIN TRAN SELECT \* FROM MLP\_OIR (HOLDLOCK) WHERE TestID <= 100

Open a new session and execute the following:

-- Session 2 ALTER INDEX [PK\_\_MLP\_OIR\_\_8CC33100059EFD65] ON MLP\_OIR REBUILD WITH ( ONLINE = ON ) GO

This query is now in blocking mode and waiting for transactions to be released from session 1. This is the default behavior before SQL Server 2012.

Second phase of experiment would be to use Lock Priority. Make sure to cancel the query from Session 2. And replace the Rebuild query as the syntax below:

-- Session 2 ALTER INDEX [PK\_\_MLP\_OIR\_\_8CC33100059EFD65] ON MLP\_OIR REBUILD WITH ( ONLINE = ON ( WAIT\_AT\_LOW\_PRIORITY ( MAX\_DURATION = 10 MINUTES, ABORT\_AFTER\_ WAIT = SELF ) ) ) GO

In this above query, we are waiting for 10 minutes for transactions to release lock for Alter Index operation. After those 10 minutes, the Alter Index will abort itself.

In the meantime, if a select is performed in Session 3, it will go through, because Shared Locks are compatible lock types.

-- Session 3 SELECT \* FROM MLP\_OIR (HOLDLOCK) WHERE TestID <= 10

If we check the sysprocesses DMV, we can find that the ALTER statement connection is on a Low\_Priority Waittype as shown below.



SELECT \* FROM sys.sysprocesses WHERE spid = 53

Additionally, if we look at the transaction locks DMV, we are presented with similar information, stating that the current connection is on a Sch-M Low\_priority\_convert state.

SELECT \* FROM sys.dm\_tran\_locks WHERE request\_session\_id = 53

Once the timeout occurs, in the above case, Session 2 will abort after 10 minutes and the below error is presented in Session 2.

Msg 1222, Level 16, State 56, Line 1 Lock request time out period exceeded.

Buffer pool Extension

Databases are a memory-hungry process. Irrespective of the type of database we choose, this option remains the same. The operating system has main memory called RAM, files on disk – contents brought to the main memory by programs, and paging file – contents pushed by the OS from main memory. This is an interesting addition because before SQL Server 2012, the buffer pools and file systems were used by the database. The buffer pool is a cache of objects from database files.

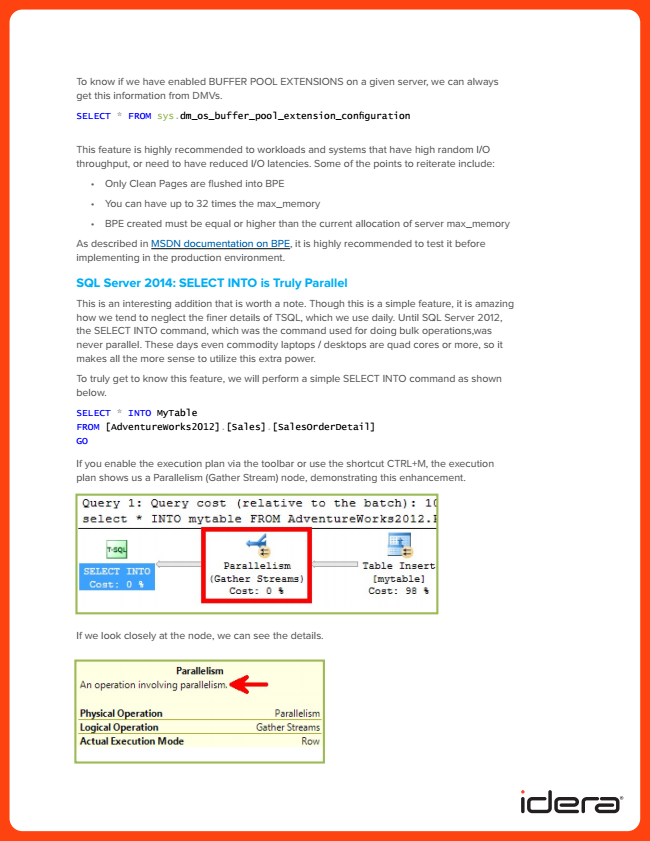
SQL Server 2014 introduces a cache to store pages that are pushed from the main buffer pool memory to a faster disk. In this case, the database pages can either be in the buffer pool, buffer pool extension file or the database files.

**Why this feature?**

The primary reason to introduce this feature was to avoid random harddisk access. With the introduction of SSD drives as an extension, these NAND flash storage drives are optimized for random access patterns. In essence, database files are stored in normal storage while the extended buffer pool is stored in SSD drives. The advantage of this is that you have a bigger buffer pool, which was not possible with a conventional RAM.

In the current release of SQL Server 2014, only clean buffer pages are allowed to be pushed into the Buffer Pool Extensions. To enable this feature, use the following T-SQL:

ALTER SERVER CONFIGURATION SET BUFFER POOL EXTENSION ON ( FILENAME = ‘D:\BufferPoolExtensions\BPE.bpe’, SIZE = 10 GB)



To know if we have enabled BUFFER POOL EXTENSIONS on a given server, we can always get this information from DMVs.

SELECT \* FROM sys.dm\_os\_buffer\_pool\_extension\_configuration

This feature is highly recommended to workloads and systems that have high random I/O throughput, or need to have reduced I/O latencies. Some of the points to reiterate include:

• Only Clean Pages are flushed into BPE

• You can have up to 32 times the max\_memory

• BPE created must be equal or higher than the current allocation of server max\_memory

As described in MSDN documentation on BPE, it is highly recommended to test it before implementing in the production environment.

**SQL Server 2014: SELECT INTO is Truly Parallel**

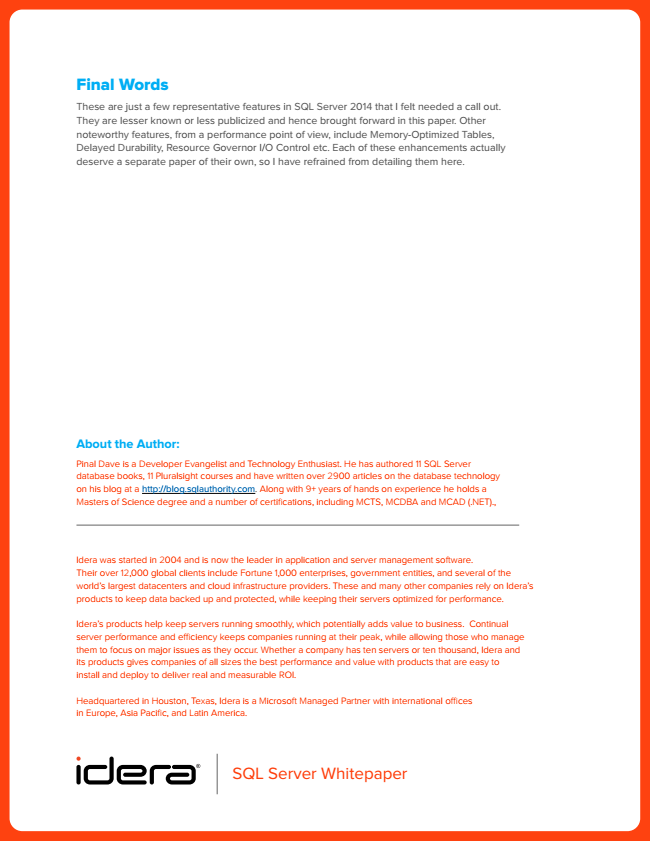
This is an interesting addition that is worth a note. Though this is a simple feature, it is amazing how we tend to neglect the finer details of TSQL, which we use daily. Until SQL Server 2012, the SELECT INTO command, which was the command used for doing bulk operations,was never parallel. These days even commodity laptops / desktops are quad cores or more, so it makes all the more sense to utilize this extra power.

To truly get to know this feature, we will perform a simple SELECT INTO command as shown below.

SELECT \* INTO MyTable FROM [AdventureWorks2012].[Sales].[SalesOrderDetail] GO

If you enable the execution plan via the toolbar or use the shortcut CTRL+M, the execution plan shows us a Parallelism (Gather Stream) node, demonstrating this enhancement.

If we look closely at the node, we can see the details.



Final Words

These are just a few representative features in SQL Server 2014 that I felt needed a call out. They are lesser known or less publicized and hence brought forward in this paper. Other noteworthy features, from a performance point of view, include Memory-Optimized Tables, Delayed Durability, Resource Governor I/O Control etc. Each of these enhancements actually deserve a separate paper of their own, so I have refrained from detailing them here.

**About the Author:**

Pinal Dave is a Developer evangelist and technology enthusiast. He has authored 11 sQl server database books, 11 Pluralsight courses and have written over 2900 articles on the database technology on his blog at a http://blog.sqlauthority.com. along with 9+ years of hands on experience he holds a Masters of science degree and a number of certifications, including MCts, MCDBa and MCaD (.net).,

Idera was started in 2004 and is now the leader in application and server management software. their over 12,000 global clients include Fortune 1,000 enterprises, government entities, and several of the world’s largest datacenters and cloud infrastructure providers. these and many other companies rely on Idera’s products to keep data backed up and protected, while keeping their servers optimized for performance.

Idera’s products help keep servers running smoothly, which potentially adds value to business. Continual server performance and efficiency keeps companies running at their peak, while allowing those who manage them to focus on major issues as they occur. Whether a company has ten servers or ten thousand, Idera and its products gives companies of all sizes the best performance and value with products that are easy to install and deploy to deliver real and measurable roI.

Headquartered in Houston, texas, Idera is a Microsoft Managed Partner with international offices in europe, asia Pacific, and latin america.

SQL Server Whitepaper